Using Apex / CPM-GOMS to Develop Human-Like Software Agents

Roger Remington Michael Matessa Michael Freed Seung Man Lee

NASA Ames Research Center
MS 262-4
Moffett Field, CA 94035
{rremington, mmatessa, mfreed, smlee} @mail.arc.nasa.gov

ABSTRACT
Here we report on the development of software agents with human-like performance characteristics for use in simulations of new airspace concepts. The goal of our effort is to use the agents to evaluate the impact of new technologies, such as widespread automation, on the workload and situation awareness of air traffic controllers. Our approach combines a task analysis, which provides a functional description of the domain, with a human performance architecture, from which detailed performance predictions can be made. We are in the preliminary stage of agent development. Here we will focus on how a standard task analysis can be paired with a human performance model to generate behavior that approximates that of the human user. We will also discuss the simulation environment and our progress to date.

General Terms
Performance, Design, Human Factors.

Keywords
GOMS, Apex, Task/User Modeling, aviation, air traffic control.

1. INTRODUCTION
Here we report on the development of software agents with human-like performance characteristics for use in simulations of new airspace concepts. The term "software agent" is used rather then "cognitive agent" since one application of our agents will be to replace some of the human participants in large-scale human-in-the-loop simulations, and it is useful to distinguish artificial from human cognitive agents. Developing agents for real-time simulations is only one goal. The specific goal of interest here is the development of agents that provide insight into the effects of new technology on air traffic controllers. More generally we are exploring the use of software agents to evaluate a human-system interaction in a range of domains. The success of efforts such as TAC-AIR Soar [12] has demonstrated the potential of using software agents as replacements for humans in simulations. Our approach departs from that of TAC-AIR Soar, and autonomous agent work in general, by integrating a model of the task with a model of the human resource architecture at the level of cognitive, perceptual, and motor (CPM) operators. The mapping of CPM operators to task actions is governed in part by theory, and in part by experimental observation. The inclusion of a fixed processing architecture at this level of detail allows our agents to capture important characteristics of human performance that remain fixed across tasks and domains. The method we describe is being applied to simulate the performance of a human operator for the purpose of evaluating the efficacy of human-system designs. In particular, the human-system designs of most interest concern human interacting with intelligent agents, both human and artificial, in systems with distributed authority. Thus, though our approach may differ from that generally taken in developing autonomous agents, our interest in the effectiveness of human-agent interaction puts us squarely within the interest profile of this workshop.

In developing human-like agents we combine a functional analysis of the task domain with a computational theory of human performance. The task analysis consists of a hierarchical task decomposition based on the Goal, Operators, Methods, and Selection (GOMS) technique [4]. The resource architecture currently implemented is based on CPM-GOMS. CPM-GOMS is a modeling method that combines a hierarchical task decomposition with a resource architecture. Tasks are decomposed in a nested set of **G**oals, **O**perators applied to transition between goal states, **M**ethods consisting of set of operators commonly applied to achieve a goal, and **S**election rules that choose between methods. The GOMS analysis terminates in low-level **C**ognitive, **P**erceptual, and **M**otor operators. CPM-GOMS models have made accurate, zero-parameter predictions about skilled user behavior in routine tasks [10]. We have recently described an approach for automatically generating CPM-GOMS analyses using the Apex computational architecture [11]. In this approach, the top-level goal is decomposed into subgoals down to the level of "templates". Templates describe elementary actions in the task domain (e.g. move-and-click a mouse, type a key). A template is a theory that describes how the underlying cognitive, perceptual, and motor operators combine to accomplish this short elementary task. Apex composes long behavioral sequences by interweaving the templates for successive templates. Many templates are common across tasks, allowing reuse of existing models and code. Composing behavioral sequences in this way from templates that describe fundamental task actions allows the model to easily simulate the overlapping of tasks characteristic of skilled human performance.

The automation of CPM-GOMS in Apex makes it practical for the first time to derive detailed predictions of human performance with complex tasks and interfaces. We have focused initially on evaluating routine human-system interaction by predicting the time and resource demands of accomplishing common interface tasks. Capturing resource demands is a crucial component of predicting how effectively displays and controls are designed, or how efficiently two concurrent tasks can be done. We will describe the application of this technique to the evaluation of human-system interfaces for pilots and controllers.

In the Virtual Airspace Modeling and Simulation (VAMS) project, a joint NASA and FAA effort, modeling and simulation methods are being applied to evaluate the effect of changes in the operation of the national airspace. New concepts, such as closer spacing of aircraft on landing, or a more distributed air transportation system, are being proposed as potential ways of increasing the capacity and safety of civil aviation. New automation under consideration would provide individual pilots and air traffic controllers more information about their situation, and help them in predicting near-future states of the airspace. VAMS is developing the modeling and simulation infrastructure that would allow the effect of these proposed changes to be evaluated. Autonomous agents provide an enabling role in two ways. First they replace humans in real-time simulations, drastically reducing the cost of simulation. Second, agents will be used in non-real-time simulations to evaluate the impact of proposed changes to airspace operations. Agents with some degree of autonomy may also come to play critical roles in actual airspace operations. Several airspace concepts involve advanced automation on the ground communicating with automated systems onboard aircraft to negotiate clearances automatically [3]. It is envisioned that human operators will interact with this technology in a supervisory role. Explorations of agent interaction may shed light on the nature of communication between human and intelligent systems.

Our agents perform the functions of aircrew or air traffic controller in non-real-time simulations of the US national airspace. A human-like agent must meet a set of functional requirements that derive from the role it will play in the simulation, and from our desire to closely match human behavior. To simulate novel airspace concepts VAMS has developed a new airspace simulation system, ACES [5]. Briefly, ACES is an agent-based event-driven simulation supporting common communications protocols (e.g. .HLA-RTI). ACES supports a simulation of a set of aircraft flying from source to destination locations across a large section of the US national airspace system (NAS), including aircraft-to-aircraft and aircraft-to-ground communications. ACES models the dynamic behavior of different aircraft types, as well as the procedures and communications agents such as aircraft and air traffic control. In its current build, ACES represents agents at a level that abstracts over individual humans and human-system interfaces. Since an important goal for us is to evaluate such interfaces, we model the entire joint human-system in Apex, and use the combined output to control an ACES agent.

The nature of human and agent interaction in airspace operations depends on the role of the agent. The crew of an aircraft acts largely as a closely-knit team, as do air traffic controllers. For these agents we can build on existing characterizations of teams [1]. In contrast, individual aircraft do not have the strong common goal with respect to each other that characterizes a team, nor are they the simple agents that characterize swarms. Aircraft are in competition with each other for airspace resources. Incorporating the proper incentive structure into agents becomes important, especially true for certain "self-separation" concepts that propose a distributed, decentralized control, where aircraft obey rules of the road and negotiate between themselves to resolve potential conflicts. Under competitive pressure certain seemingly obvious rules of the road have been shown to lead to suboptimal outcomes [6]. We build on existing "belief" representations [2] to model these behaviors.

The nature of airspace operations dictates capabilities in an autonomous agent needed to produce the human-like behavior required to evaluate proposed concepts. Skilled human action reflects the attempt to respond to environmental demands subject to processing limitations. The characteristics of the environments we are interested in require the following capabilities:

- Domain knowledge that dictates when to take action, what action to take, and how to control dynamic objects (e.g. aircraft), including rules of conversation that dictate when and how to communicate
- Common strategies for dealing with nominal and off-nominal conditions
- Multitasking capability that supports interruption, task suspension, task resumption, and task interleaving
- A representation of "beliefs" or "expectations" about what other agents in the simulation are doing, or likely to do [2], including the planning of future behaviors
- Limitations in human information processing require that the agent be capable of reflecting human resource management. In particular, it requires the following:
- A set of resources and a rules that determine how they interact
- A set of parameters characterizing normal latencies and variability
- Theory or heuristics that map resources onto activities in the task domain

The requirements for environmental action and resource management are well matched to the capabilities of the Apex computational architecture, developed at NASA Ames Research Center [8, 9]. Apex models an agent attempting to schedule its limited resources to accomplish multiple tasks. Apex implements a reactive plan execution mechanism [8, 9], and a language capable of expressing a range of multitasking functionality including interruption, suspension, and resumption.

In the following sections we describe in detail our modeling approach, explain how that approach can be included in the ACES airspace simulation framework, and finally show the progress of our modeling efforts for an air traffic control handoff task.

2. THE MODELING APPROACH

Our approach to this has been to develop a cognitive modeling method that allows us to compose behavior from elementary human Cognitive, Perceptual, and Motor operators, whose characteristics are relatively constant across domains. Key to our success with the compositional approach has been the development of a method for automatically scheduling these elementary human resources, which was achieved using the Apex computational architecture. A high-level overview of an Apex model is shown in Figure 1.

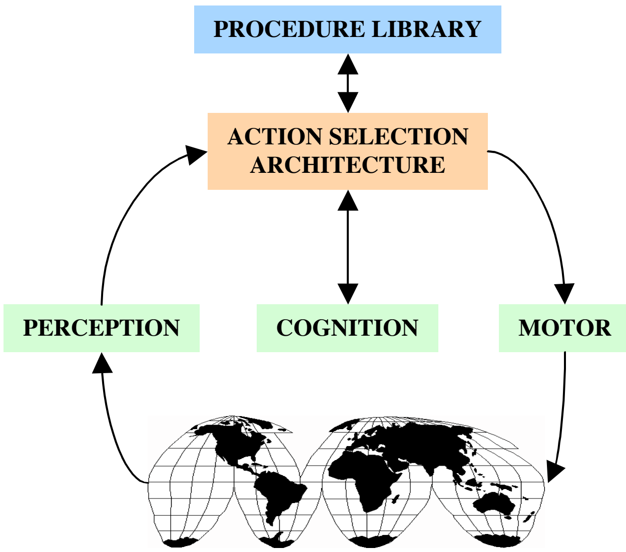

Figure 1: The Apex modeling architecture

The modeling framework includes a ***Resource Architecture***, an ***Action Selection Architecture***, and a ***Procedure Library***. The resource architecture defines the limited-capacity cognitive, perceptual, and motor components as shown in Figure 1. In actuality these are categories of resources and a running model would generally include a more detailed specification within each. The action selection architecture coordinates the activity of the resources and applies knowledge in the form of procedures. The procedure library contains the knowledge the agent applies to perform in the target domain. All knowledge in Apex is in the form of procedures. Information about the world is input through perceptual processes, which have limited capacity. Incoming information is matched against the specifications of procedures in the procedure library. If conditions for a procedure are detected then the Action Selection Architecture schedules the steps of the procedure in accordance with the constraints specified below.

As discussed earlier the model combines the CPM-GOMS human performance model with the Apex computational architecture, which provides the underlying simulation framework. Apex [8, 9] is a software tool for creating, running and analyzing simulations of intelligent agents carrying out human-computer interaction tasks. Apex treats the problem of modeling behavior as a problem of scheduling an agent's limited resources. The agent architecture incorporates a reactive planning and execution mechanism [8, 9] with integrated online resource scheduling and other capabilities needed to handle multiple tasks. Both the general capabilities of a reactive planner and the multitask management extensions specific to Apex have proven central in automating CPM-GOMS models.

The reactive planner recursively decomposes high-level goals into subgoals and primitive operators based on stored plans. What makes the planner reactive is that it does not generate the goal hierarchy all at once. Instead, it waits until all preconditions associated with a given goal are satisfied before retrieving a plan to specify subgoals. Deferring goal decomposition until near execution time enables the planner to choose how to decompose (i.e. which procedure to use) with as much situation information as possible. This strategy is essential for tasks such as air traffic control where uncertainty about future world state is high and the need for careful deliberation, either to solve complex problems or to choose optimally from a wide range of possible solutions, is low. The reactive planner allows the Apex controller agent to be interrupted by voice communications or by alerts that signal the need for immediate action.

The process of recursively applying methods to non-primitive goals (those that do not correspond to an operator) produces a goal hierarchy, consistent with GOMS. In a standard GOMS analysis this recursion would bottom out in leaf nodes in the task domain, such as depicted in Figure 4. Our method of behavior composition has the task hierarchy bottom out in a set of primitive ***templates***. These templates describe elementary task behaviors, such as moving and clicking a mouse, or typing a key, that occur in many contexts. Thus, by modeling these elementary behaviors we can develop a library of behaviors from which to construct larger behavioral sequences. In this way, we bridge the gap between the standard task analysis and a user model of some detail. By separating these two elements we hope to free the designer from specifying the cognitive architecture, and allow her to execute an agent model from leaf nodes in the task domain.

There are two issues that arise in this compositional approach: how templates are constructed, and how they are combined to produce extended behavioral sequences. We first discuss briefly how templates are constructed, then how they are combined to model larger agent behavioral sequences.

2.1 Template Construction

In CPM-GOMS the leaf nodes of the hierarchy form a sequence of operator-level actions composed of elementary cognitive, perceptual, and motor activities. Under strong assumptions about operator independence – that operators are executed in strict sequence and that the specific nature of an operator has no effect on the time required to execute other operators in the sequence – the total task time would be the sum of these leaf node operators. These assumptions do not hold for highly practiced sequences where the execution of adjacent operators may overlap in time and the degree of overlap can depend on operator order and identity. As a result, the sum duration of operators whose assigned individual durations are correct in isolation will tend to predict excessive overall task duration.

Instead, CPM-GOMS allows parallel operator execution subject to three kinds of constraints. **Logical constraints** apply when one action is required to specify or enable another and therefore must precede it. Logical constraints may apply between operators within a single template or across templates. Within-template constraints must be specified in the template definition. Since CPM-GOMS allows operators from one template to slip back into the temporal scope of earlier templates, making these constraints explicit and including mechanisms to enforce them would be required to guarantee correct behavior.

Unary resource constraints specify that when two operators use a non-sharable, non-depletable resource (e.g. the left-hand), they cannot be carried out concurrently. Within a template, operators requiring the same resource must be explicitly sequenced. Across templates, operators must follow a template precedence rule such that the highest priority task gains access to resources.

Slack exclusion constraints are restrictions on the use of slack time by operators with specified properties. Slack exclusion constraints are often applied on operators representing a cognitive action to initiate a motor response. The rule for a cognitive initiation exclusion rule can be defined as follows: given a set of operators $A_1..A_m$ from template t_i that use resource R, operators B and C from template t_j where C uses R and B represents a cognitive action that initiates C, and i<j, B cannot execute until all A_k are complete. This rule contributed to very accurate predictions in the referenced model; however, its generality and scientific basis are still undetermined.

2.2 Template Interleaving

The same parallelism that applies to elementary operators within a template applies to operators across templates. Thus, it is generally not possible to obtain accurate predictions simply by concatenating templates. We have previously reported [11] a method of interleaving the elements of successive templates that provides excellent fits to observed data. The details of this method are beyond the scope of this paper, but in brief, interleaving of templates is accomplished by the application of the three constraints described above. We have successfully fit data from human-computer interaction tasks including mouse-based automated teller simulations, typing, and computer aided design tasks. The method is currently being applied to shuttle cockpit procedures, airline cockpit procedures, and as here air traffic control operations. Figure 2 shows the fit between observed and predicted data for the automated teller machine task.

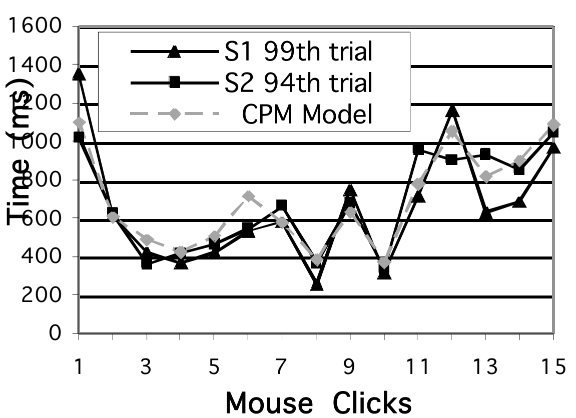

Figure 2: Apex model fit for automated teller machine task

In summary, our compositional approach to simulating behavior relies on a theory of resources to predict concurrency, and on a software architecture to execute reactive plans. Concurrent operator execution occurs within and across templates. Within-template concurrency arises because cognitive, perceptual, and motor operators call on separate, independent resources. Between-template concurrency arises from the interleaving of operators from different templates. The essence of the interleaving phenomenon is that the activities specified by a template do not use all resources all of the time; idle time (slack) in the use of a resource by one template's operators represents an opportunity for operators from a later template to "slip back" and begin execution. Interleaving at the level of CPM-GOMS operator-level goals corresponds to overlap in the execution of higher-level goals – i.e. at the level of templates or classic GOMS operators – and thus accounts for the different predictions of the CPM-GOMS and classic GOMS approaches.

That all knowledge is represented as procedures raises the issue of how to deal with representational issues. For example, in air traffic control it may be necessary to deal differently with "beliefs" than with knowledge of which one is more certain. We have yet to deal with this issue directly. However, Apex includes mechanisms for adjusting the priority of procedures based on current information. It may be that an Apex agent will hold several different beliefs and alter its beliefs by selecting the appropriate one for the given circumstance.

Previous Apex models have all used a simulated environment that was internal to Apex. In order to allow Apex models to participate in large-scale simulations of the national airspace system at NASA, we are integrating Apex into the ACES simulation environment.

3. THE SIMULATION ENVIRONMENT

The simulation environment, ACES, was introduced above and briefly described. One of the challenges in working with external simulation environments is to synchronize the timing of the model with the simulation. Our Apex models achieve accurate predictions by simulating performance at a granularity of about 50 milliseconds. It is necessary to represent human behavior at such a fine granularity in order for the interleaving to accurately reflect the parallelism evidenced in performance. Because the ACES simulation is event driven, each agent is allowed to update its state at whatever granularity is appropriate. However, ACES agents are typically large-scale airspace entities, such as aircraft and air traffic control centers. It would be inefficient to simulate the large-scale flow of traffic at a fine granularity. Indeed, the simulation has a default granularity of 1 second, 20 times slower than the human performance model.

Figure 3 shows our initial attempt at accommodating this difference. Apex will be used to model both the human agent and the displays and controls the agent interacts with. In this way, timing of agent actions and delays imposed by the equipment and user interface can be simulated at high fidelity. Communication with the ACES simulation will occur at synchronized message passing times in accordance with ACES protocols. This arrangement is depicted in Figure 3. This scheme provides a natural division between the human performance model and the ACES-level agent. ACES incorporates objects that represent high-level agents. The Apex agent need only transmit to its ACES counterpart those messages of significance in the larger context. Thus, while each keystroke of data entry into flight computers must be simulated to predict the time, the ACES-level agent need only be informed when the keystrokes produce some change in the its state, such as activating a mode, or changing a control setting.

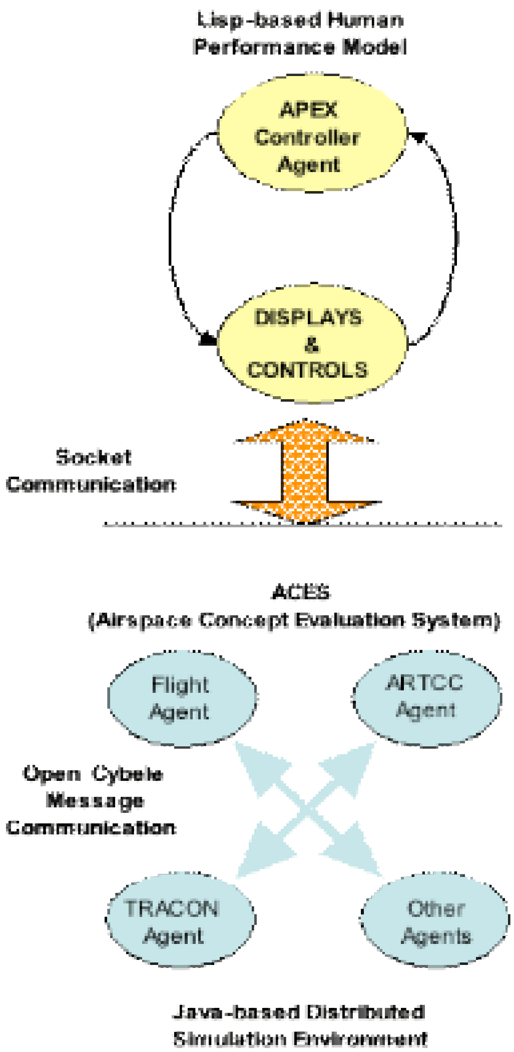

Figure 3: Integrating Apex with ACES

Timing issues could still arise as delays in the simulation due to Apex computations. Such delays should be the result of Apex delays only if the requisite computations cannot be performed within the 1-second update interval. To date, achieving this level of performance has not been difficult. The key constraint on Apex time is the range of tasks over which interleaving occurs, as this defines the complexity of arranging and rearranging the agenda. The Apex language contains expressions that allow us to limit the scope of interleaving

4. THE TASK

At present, ground based air traffic control installations have responsibility for routing of aircraft and maintenance of safe separation between aircraft. There are three principle types of installation: terminal, TRACON, and enroute. Terminal control handles surface operations including take-off and landing. TRACON handles arrival and departure sectors for airports. Enroute handles the remainder of the flight between departure and arrival sectors. Other entities modeled in the ACES simulation include the airline operating centers, which are run by each airline and provide schedule and gate information to their own aircraft, and the aircraft themselves. Previous work has attempted to provide a more or less complete functional analysis of TRACON [14] and enroute [5]. Those have not yet been turned into workable computational agents. Some progress toward a complete controller agent has begun [2], but the controller's task is a complex mixture of spatial, geometric, temporal, and procedural reasoning. No software agent has yet been able to handle more than a portion of the entire task though progress in applying rule-based systems has yielded agents with some capability [2, 5].

We have begun our agent development by focusing on enroute control, following existing task analyses [5]. Figure 5 presents a high level functional analysis of a portion of current enroute control, taken with small modifications from [5]. Each sector is managed by two controllers, the radar controller (R-side) and the radar associate controller (D-side), whose duties complement and overlap. The R-side controller is the primary controller in contact with aircraft. The D-side controller assists, sharing responsibility for conflict detection and planning, as well as performing routine housekeeping. For the present discussion we will focus on the R-side controller activities involved in initiating a handoff. To estimate the nature of the interaction among the various agents in the airspace simulation, we modeled the communications required to accomplish a handoff, the lighter text in the section of Figure 4.

It is apparent from Figure 4 and the earlier discussion that an individual controller is part of a network of airspace entities. Each individual shares responsibility for ensuring safe operations, but those responsibilities are themselves divided. For example, the aircrew is primarily responsible for the safety of the their aircraft, the controller for the safety of all aircraft in her sector. In addition to the formal rules for controllers (as well as aircrew) there exist informal practices. These informal rules have not been exhaustively studied, but are apparent in the behavior of controllers. They represent social contracts between entities that have emerged with time. Because of reciprocity, many informal rules act for the general good of all players. For example, controllers will try to manage the traffic in their sector so that they do not overburden the downstream controller to whom they hand the traffic off. This cooperative behavior may not apply if they are busy.

More formal contracts also apply. It is understood that and aircraft will respond relatively quickly to a command by a controller, which they have acknowledged. This quick response reduced controller workload. Until evidence of the maneuver appears on the controller's radar screen, the controller has only the "belief" that the agent will carry out the instruction. The maintenance of such beliefs takes memory resources, which are reduced by reducing the time over which they need apply. This is an important consideration since new datalink equipment will eliminate the vocal transmission and acknowledgment characteristic of present practices. One of the goals for our agents is to shed light on how new equipment and practices affect controller interactions with aircraft and other controllers.

The light text in Figure 4 represents more detailed functional steps in initiating a handoff. We developed a simple model of this procedure in Apex using observed times for the interaction of the various agents involved [5].

- Radar Controller (R-side)
  - Handoff aircraft (a/c)
  - Receive handoff
  - Resolve conflicts
  - Resolve metering/spacing violations
  - Respond to Alarms
  - Provide weather avoidance maneuvering
  - Provide flight plan conformance
  - Communicate clearances
  - Accommodate requests
  - Communicate with descent sectors
  - Receive Status Updates
  - Housekeeping
  - Monitor Situation
    - Initiate handoff transfer
      - Acquire SA for AC
      - Determine Response
      - Reject
      - Accept
        - Detect Acceptance
        - Issue Freq Change
        - Receive Feedback
        - Mark AC as Shipped
    - Detect Violation
    - Detect Top of Descent
    - Detect Conflicts
    - Determine weather impact on a/c
    - Monitor radio
- Radar Associate Controller (D-side)
  - Perform conflict detection
  - Perform trial planning
  - Coordinate with other sectors
  - Assist with pilot clearance readback
  - Manage a/c list and flight data
  - Enter flight plan amendments

Figure 4: Functional analysis of enroute control

5. PROGRESS

The model output, shown in Figure 5, highlights the communications overhead associated with a routine sector operation. Each colored row in Figure 5 represents a single agent in the airspace simulation. The boxes in the top row depict the times for the R-side controller receiving the handoff. The second row depicts the transferring controller on the R-side. The third row represents the aircrew response. The final row the D-side activities on the transferring side. A key observation is the presence of slack in the activity schedule caused by waiting for the transfer activities and the aircrew response. New equipment currently in field test would eliminate this slack by allowing the controller to transfer and receive handoffs automatically. The technology, termed datalink, is a direct connection from ground computers to flight computers on board the aircraft. One of the questions is how the new technology will impact operations. As seen above, eliminating the waiting will reduce the slack and save time and hopefully workload. However, the success of such devices is often in the details of the human-computer interface. To fully evaluate this technology it will be necessary to develop a more detailed agent model that interacts with the target device.

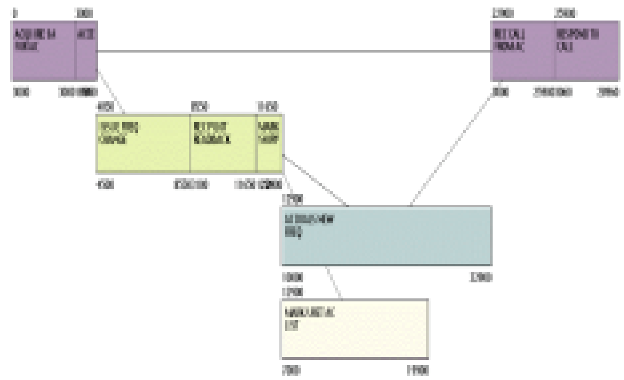

Figure 5: Timing output of Apex model of handoff transfer

The model output of Figure 5 reflects the interaction among agents in accomplishing a task. Parameters for the model were taken as measurements in human-in-the-loop airspace simulations. We would like our agents to exhibit human-like performance characteristics with a variety of interfaces and rules that reflect changing concepts for airspace operations. In many cases prototype equipment will not exist so such direct measurements would not be possible. Indeed, it would be useful if the agent could help define procedures and information displays that would promote efficient human-machine communication.

6. ACKNOWLEDGMENTS

This work was supported by the Virtual Airspace Modeling and Simulation project of the NASA Airspace Systems program. The authors wish to thank Lisa Bjarke, Karlin Roth, and Larry Meyn for programmatic assistance, and Parimal Kopardekar for making available task analysis and task modeling data.

7. REFERENCES

[1] Beavers, G & Hexmoor, H. Teams of agents. In *Proceed. 2001 IEEE Man & Cybernetics Conference*.

[2] Callantine, T. CATS-based Air Traffic Controller Agents. NASA Contractor Report 2002-211856, 2002.

[3] Callantine, T., Prevot, T., Smith , N., & Palmer, E. Simulation of CTAS/FMS Air Traffic Management. Proceed. 4th USA/Europe Air Traffic Management Research & Development Seminar, Santa Fe, NM, 2001.

[4] Card, S., Moran, T., & Newell, A. (1983) The Psychology of Human-Computer Interaction. Lawrence Erlbaum Associates, Hillsdale, NJ.

[5] Computer Sciences Corporation. Single-Year, NAS-wide benefits assessment of DAG-TM Ces 5, 6, and 11. ATMSDI contract report for CTOD-8 NAS2-00014.

[6] Erev, I., Barron, G., & Remington, R. Right of way in the sky: Two proble ms in aircraft self-separation and the auction-based solution. *Human Factors* (submitted).

[7] Firby, R. J. Adaptive Execution in Complex Dynamic Worlds. Ph.D. thesis, Yale University, Computer Science Department. Technical Report 672, 1989.

[8] Freed, M. Simulating Human Performance in Complex, Dynamic Environments. Doctoral Dissertation, Northwestern University.

[9] Freed, M., Matessa, M., Remington, R., & Vera, A. (2003). How Apex automates CPM-GOMS. Presented at annual meeting of the International Conference on Cognitive Modeling, Bamburg, Germany, April 2003.

[10] Gray, W. D., John, B. E. & Atwood, M. E. (1993) Project Ernestine: Validating a GOMS Analysis for Predicting and Explaining Real-World Task Performance*, Human-Computer Interaction,* 8, pp.237-309.

[11] John, B., Vera, A., Matessa, M., Freed, M, & Remington, R. Automating CPM-GOMS. *Proceed. CHI, 2002* Minneapolis, MN, April 20-25.

[12] Laird, J.E., Coulter, K.J., Jones, R.M., Kenny, P.G., Koss. F., & Nielson P.E. Automated Intelligent Pilots for Combat Flight Simulation, *AI Magazine* pp 27-42.

[13] Raytheon ATMSDI Team. Creation of a modeling and simulation capability to support NAS-Wide analyses of advanced ATM tools and concepts. NASA Contractor Report CTOD 7.18-2, Contract #NAS2-00015, 2002

[14] Rodgers, M.D. & Drechsler, G.K. (1995). Conversion of the TRACON operations concepts database into a formal sentence outline job task taxonomy. DOT/FAA/AM-95/16.